Süsteemprogrammeerimine keeles C

11 Loeng

UNIX ja failid

- Kogu arvuti on UNIXi jaoks failid
  - Fail on fail
  - Kataloog on fail
  - *FIFO special* (e. *named pipe*) on fail
  - *Character special* on fail (*device file alamtüüp*)
  - *Block special* on fail (*device file alamtüüp*)
  - *Symbolic link* on fail

Faili atribuudid

Attribute	Value meaning
File type	Type of file (regular, directory, fifo, ...)
Access permission	File access permissions for different users
Hard link count	Number of hard links of a file
UID	User ID of file owner
GID	Group ID of file
File size	File size in bytes
Last access time	Time the file was last accessed
Last modification time	Time the file was last modified
Last change time	Time the file attribute was last changed
Inode number	Inode number of the file
File system ID	File system ID where the file is stored

NB: Failinimi ei ole faili atribuut!

Inode

- Faili kohta käivat infot hoitakse struktuuris, nimega *inode*.
- Reeglina hoiab failisüsteem neid massiivis, mida nimetatakse *inode table*.
- *Inode*'i number on failisüsteemis iga faili jaoks unikaalselt identifitseeriv

Kataloogid

- UNIXi kataloogid on failid, mis sisaldavad paare:
  - inode:nimi
- Kataloogis on ka failid "." ja "..", mis viitavad vastavalt kataloogi enda *inode*'ile ja üks tase kõrgema kataloogi *inode*'ile. (V.a juurkataloogi "/" puhul, kus ".." viitab samuti iseendale)
- Kui inode on 0, siis on vastav koht kirjutamiseks vaba

File descriptor

- UNIXis toimub kogu sisend ja väljund failidesse kirjutamise ja failidest lugemise abil: kõik lisaseadmed, sealhulgas klaviatuur ja ekraan on süsteemis olevad failid.
- Faili kirjeldab protsessi jaoks väike positiivne täisarv
- Mõnikord kasutatakse ka terminit kanal *(channel)* – failideskriptor määrab kanali

Eeldefineeritud failideskriptorid

- Süsteem avab programmi jaoks kolm failideskriptorit:
  - 0 standard input (stdin)
  - 1 standard output (stdout)
  - 2 standard error (stderr)
- Enamasti on deskriptorid programmi failitabeli indeksid

Kanali loomine

```
– int open(const char *pathname, int flags);
– int open(const char *pathname, int flags, mode_t
  mode);
– int creat(const char *pathname, mode_t mode;
– int close(int fd);
```

- creat() avab faili kirjutamiseks: sisuliselt open lippudega O_CREAT|O_WRONLY|O_TRUNC
- Mode on failisüsteemi mode'i loabitt (0640 on kasutajale RW ja grupile R, teistele keelatud)
- Flag on O_RDONLY, O_WRONLY või O_RDWR kombineerituna O_APPEND, O_CREAT ja O_EXCL jne.

Stream kanalist

```
FILE *fdopen (int fd, const char *mode);
```

- Muudab olemasoleva failideskriptori *stream*iks.
- Mode on "r", "w" jne nagu fopen() funktsioonis

```
int fileno(FILE *stream);
```

- Tagastab avatud streami failideskriptori

Sisend/Väljund

```
size_t read(int fd,  void *buf,  size_t count);
size_t write(int fd,  const void *buf, size_t count);
```

- Tagastavad mitu baiti liigutati
- Liigutada võib üsna suvalise arvu baite, kuid levinud väärtused on 1, mis oleks puhverdamata kirjutamine, või 1024 või 2048, mis võivad vastata välisseadme bloki suurusele. Suurem number on reeglina efektiivsem, sest nii tehakse vähem süsteemikäske
- Puhverdamist ei ole (see toimub mõnel madalamal tasemel sellegipoolest)

Näide: getchar.c

```
/* getchar1.c */


#define EOF (-1)
#define STDIN   0
#define STDOUT 1
#define STDERR 2

int getchar() {
    char c;
    if (read(STDIN, &c, sizeof(c))==0)
        return EOF;
    else
        return c;
}
```

Näide: getchar2.c

```
/* getchar2.c:  lihtne puhverdatud versioon */
   int getchar(void)
   {
       static char buf[BUFSIZ];
       static char *bufp = buf;
       static int n = 0;

       if (n == 0) {  /* puhver tühi */
           n = read(0, buf, sizeof (buf));
           bufp = buf;
       }
       return (--n >= 0) ? (unsigned char) *bufp++ : EOF;
   }
```

Suvapöördumine (Random access)

```
off_t  lseek(int fd,  off_t offset,  int whence);
```

- off_t on kas long või spetsiaalne 64 bitine struktuur
- whence on (varem kasutati numbreid 0, 1, 2):
  - SEEK_SET (0) – mitu baiti faili algusest
  - SEEK_CUR (1) – mitu baiti praegusest positsioonist
  - SEEK_END (2) – mitu baiti lõpust
- Edu korral tagastatakse uus off_t

Kanali kloonimine

```
int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

- Pärast kloonimist võib failideskriptoreid kasutada vaheldumisi: nende failipositsioonid, ligipääs ja lipud on jagatud. Kui ühe positsiooni muuta, muutub ka teine.
- dup() tagastab kõige väiksema parajasti vaba oleva deskriptori numbri
- dup2() sulgeb vajadusel newfd
- tagastatakse uus deskriptor, vea korral -1
    - vea korral saab ka errno väärtuse

Näide: dup()

```
#define ERROR (-1)
#define STDIN 0

/* loeme stdin alt, kui inputfileoption on määratud,
loeme failist */

if (inputfileoption) {
   if ((ifd = open(file, O_RDONLY)) == ERROR)
      return ERROR;
   else if (close(STDIN) == ERROR)
      return ERROR;
   else if (dup(ifd) == ERROR) /* madalaim fd on 0!*/
      return ERROR;
   else if (close(ifd) == ERROR)
      return ERROR;
}
/* stdin lugemine loeb nüüd faili */
```

Hardlink

```
int link(const char *oldpath, const char *newpath);
```

- link() teeb uue lingi (*hardlink*) oldpath nimelisele failile.
- Kui newpath on olemas, seda üle ei kirjutata
- Uus fail on igas mõttes vanaga identne. Mõlemad failid viitavad ühele ja samale *inode*'ile.
- Edu korral tagastub 0, vea korral -1 ja errno väärtustub

Softlink

```
int symlink(const char *oldpath, const char
  *newpath);
```

- Loob sümboolse lingi (*Symbolic* või *softlink*)
- Sümboolne link võib olla:
  - Suhteline: ../text.txt
  - Absoluutne: /home/irve/text.txt

unlink()

```
int unlink(const char *pathname);
```

- Muudab pathname'iga viidatud faili inode'i vastavas kataloogis 0-ks, vähendab viidete loendurit vastava *inode*'i atribuutides ja vabastab andmeblokid, kui loendur 0 peale jõuab.
- PS! Tegelikkuses vabastuvad *inode* ja andmeblokid alles siis, kui viimane faili kasutav protsess oma töö lõpetab. Võid avada ajutise faili, selle unlinkida, kuid ikka selle sisu poole pöörduda.

Faili seisund

```
int stat(const char *file_name, struct stat *buf);
int fstat(int fd, struct stat *buf);
int lstat(const char *file_name, struct stat *buf);
```

- Tagastab *inode*'i kohta käiva info. Struktuur stat on defineeritud failis sys/stat.h
- fstat() on analoogne stat()-le, kuid võtab esimeseks argumendiks failideskriptori; seda siis faili nime asemel
- lstat() on stat()-ga identne selle erinevusega, et sümboolse lingi puhul tagastab lingi enda, mitte lingitava faili andmed

Stat struktuur

```
#include <sys/stat.h>, <sys/types.h>
struct stat
{
    dev_t         st_dev;      /* device */
    ino_t         st_ino;      /* inode */
    mode_t        st_mode;     /* protection */
    nlink_t       st_nlink;    /* number of hard links */
    uid_t         st_uid;      /* user ID of owner */
    gid_t         st_gid;      /* group ID of owner */
    dev_t         st_rdev;     /* device type (if inode device) */
    off_t         st_size;     /* total size, in bytes */
    unsigned long st_blksize;  /* blocksize for filesystem I/O */
    unsigned long st_blocks;   /* number of blocks allocated */
    time_t        st_atime;    /* time of last access */
    time_t        st_mtime;    /* time of last data modification */
    time_t        st_ctime;    /* time of last inode change */

};
```

stat struktuuri bittide muutmiseks on defineeritud hulgaliselt makrosid

Ligipääsu kontrollimine

```
int access(const char *pathname, int mode);
```

- access() kontrollib, kas protsessil oleks õigus viidatud faili (või muud failisüsteemi objekti) kirjutada, lugeda või selle olemasolu kontrollida.
- mode on bitimask õigustest R_OK, W_OK, X_OK, ja F_OK (viimane siis faili olemasolu)
- Tagastab 0, kui antud tegevus on lubatud, -1 muul puhul koos errno väärtustamisega

Ligipääsu määramine

```
int chmod(const char *path, mode_t mode);
int fchmod(int fildes, mode_t mode);
```

- Määrab nime või failideskriptoriga viidatud faili ligipääsuõigused
- mode on mitme biti omavahelise OR-imise tulemus:

```
#define S_IRWXU 0000700    /* RWX mask for owner */
#define S_IRUSR 0000400    /* R for owner */
#define S_IWUSR 0000200    /* W for owner */
#define S_IXUSR 0000100    /* X for owner */
#define S_IRWXG 0000070    /* RWX mask for group*/
#define S_IRGRP 0000040    /* R for group */
#define S_IWGRP 0000020    /* W for group */
/* jne... */
```

Ligipääsu määramine

```
int chown(const char *path, uid_t owner, gid_t group);
int fchown(int fd, uid_t owner, gid_t group);
```

- Muudetakse nime või failideskriptoriga määratud faili omanik ja/või grupp.
- Omanikuvahetust saab läbi viia vaid juurkasutaja.
- Kasutaja saab määrata oma failide kuuluvust gruppidele, kuhu ta ise kuulub
- Juurkasutaja võib faili grupi määrata oma suva järgi

Loodavate failide õigused

```
mode_t umask(mode_t mask);
```

- umask määrab loodavate failide õigused

  umask = mask & 0777

- umaskis püstised bitid võetakse faili loomisel *ära*.
- loabitid = mode & ~(umask)
- kui levinud vaikimisi umask on 022, mille tulemusena luuakse failid õigustega 0666 & ~022 = 0644 = rw-r--r--
- see funktsioon õnnestub alati. tagastatakse umaski eelmine väärtus

Kataloogide haldus

```
int mkdir(const char *path, mode_t mode);
```

- mkdir() teeb path argumendiga viidatud kohta uue kataloogi

```
int rmdir(const char *path);
```

- rmdir() funktsioon eemaldab kataloogi kohas path.
- Kataloog peab olema tühi st tohib sisaldada vaid . ja .. faile.

Kataloogide vahetamine

```
int chdir(const char *path);
int fchdir(int fildes);
```

- chdir() vahetab töökataloogi kas etteantud kataloogiks või failideskriptori poolt viidatud kataloogiks. Tegu peab olema kataloogiga

```
char *getcwd(char *buf, size_t size);
```

- Tagastab pointeri töökataloogile
- size peab kataloogi nimest ühe võrra pikem olema
- Kui buf pole NULL, pannakse nimi buf poolt viidatud aadressile. NULL puhul võetakse size jagu mälu ja poetatakse nimi sinna.

Kataloogi sisu vaatlemine

- Kataloogide sisu vaatlemiseks tuleb kataloog avada, sealt ükshaaval sissekanded ära lugeda ja siis kataloog sulgeda.

```
#include <sys/types.h>
#include <dirent.h>
DIR *opendir(const char *dirname);
```

- Avab dirname poolt viidatud kataloogi *stream*'i, stream. Positsioneeritakse see esimese sissekande juurde.

```
int closedir(DIR *dirp);
```

- sulgeb kataloogi *stream*'i

Kataloogi sisu vaatlemine (2)

```
struct dirent *readdir(DIR *dirp);
```

- Tagastab struktuuri dirent pointeri, mis sisaldab viidatavas kataloogi*stream*is parajasti järjekorras olevat sissekannet.
- Kataloogi lõppu jõudes tagastub NULL pointer
  - <dirent.h> fail kirjeldab kataloogisissekannet
  - readdir() kirjutab eelmise väljakutse poolt tagastatud andmed üle
  - POSIX standardi kohaselt on dirent struktuuris väli char d_name[], määramata pikkusega, maksimaalselt NAME_MAX tähemärgist, lõpetatud null märgiga. Teiste väljade kasutamine ei ole porditav. Tihti on ka d_namelen väli nime pikkusega

Näide: Nimeotsing

```
len = strlen(name);
dirp = opendir(".");
while ((dp = readdir(dirp)) != NULL)
     if (dp->d_namlen == len
                && !strcmp(dp->d_name, name)) {
      closedir(dirp);
      return FOUND;
      }
closedir(dirp);
return NOT_FOUND;
```

Asupaik kataloogis

```
void seekdir(DIR *dirp, long int loc);
```

- Määrab readdir() käsu jaoks positsiooni kataloogis. Ette anda kataloogipointer ja positsioon

```
long int telldir(DIR *dirp);
```

- telldir() tagastab kataloogi*stream*i positsiooni
- kui viimane kataloogil tehtud operatsioon oli seekdir(), tagastab telldir() sama väärtuse, mis oli seekdir() argument.

Kataloogides möllamine

```
#include <ftw.h>
int ftw(const char *path, int (*fn)  (const  char  *,
  const struct stat *, int), int depth);
```

- ftw() käib rekursiivselt pathiga näidatud kataloogi ja selle alamkataloogid läbi, kutsudes iga sissekande puhul välja etteantud funktsiooni.
- Funktsioon saab argumentideks failinime, objekti stat struktuuri, ja täisarvu:
  - FTW_F - normal file, FTW_D – directory, FTW_DNR - directory that cant be read, FTW_SL - symbolic link, FTW_NS - file that cant be stat'd
- depth on paralleelselt avatud kataloogide arv. Ületamisel suletakse varasemad ja ftw() aeglustub

Näide: Kataloogis möllamine

```
/*  We start with the appropriate headers and then a
function, printdir, which prints out the current directory.
It will recurse for subdirectories, using the depth parameter
is used for indentation.  */

#include <unistd.h>
#include <stdio.h>
#include <dirent.h>
#include <string.h>
#include <sys/stat.h>

void printdir(char *dir, int depth)
{
    DIR *dp;
    struct dirent *entry;
    struct stat statbuf;

    if((dp = opendir(dir)) == NULL) {
        fprintf(stderr,"cannot open directory: %s\n", dir);
        return;
    }
```

Näide: Kataloogides möllamine (2)

```
chdir(dir);
    while((entry = readdir(dp)) != NULL) {
        stat(entry->d_name,&statbuf);
        if(S_ISDIR(statbuf.st_mode)) {
            /* Found a directory, but ignore . and .. */
            if(strcmp(".",entry->d_name) == 0 ||
                strcmp("..",entry->d_name) == 0)
                continue;
            printf("%*s%s/\n",depth,"",entry->d_name);
            /* Recurse at a new indent level */
            printdir(entry->d_name,depth+4);
        }
        else printf("%*s%s\n",depth,"",entry->d_name);
    }
    chdir("..");
    closedir(dp);
}
```

Näide: Kataloogides möllamine (3)

```
/*  Now we move onto the main function.  */

int main(int argc, char* argv[])
{
    char *topdir, pwd[2]=".";
    if (argc != 2)
        topdir=pwd;
    else
        topdir=argv[1];

    printf("Directory scan of %s\n",topdir);
    printdir(topdir,0);
    printf("done.\n");

    exit(0);
}
```

Seadmete juhtimine

```
int ioctl(int fd, int request, ...)
int ioctl(int fd, int request, char *argp)
/* traditsioonilisem versioon */
```

- ioctl() saab saata avatud failideskriptoritele numbrilisi signaale (ja lisaargumente). Paljusid seadmeid on võimalik ioctl abil juhtida
- sys/ioctl.h sisaldab antud käskude jaoks makrosid ja define konstante.
- Abiks: man ioctl_list

Kui see slaid on ekraanil, on loeng läbi